2014 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery

Semantic-based Publish/Subscribe for M2M

Mário Antunes
Instituto de Telecomunicações
Universidade de Aveiro
Aveiro, Portugal
Email: mario.antunes@av.it.pt

Diogo Gomes
Instituto de Telecomunicações
Universidade de Aveiro
Aveiro, Portugal
Email: dgomes@av.it.pt

Rui Aguiar
Instituto de Telecomunicações
Universidade de Aveiro
Aveiro, Portugal
Email: ruilaa@av.it.pt

***Abstract*—The number of connected devices is expected to soar in the coming years, each one of them collects and distributes real-world information though various systems. As the number of such connected devices grows, it becomes increasingly difficult to store and share all these new sources of information. Several context representation schemes try to standardize this information, however none of them have been widely adopted. Publish/ subscribe paradigm has proven to be an adequate abstraction for large scale information dissemination, but none of current variations is well suited for context information. In a previous publication we addressed these challenges, however our solution has some drawbacks: poor scalability and semantic extraction. The aim of this paper is twofold. First, we discuss an efficient way to deal with representation schemes diversity and propose a d-dimensional context organization model. Second, we propose a semantic-based publish/subscribe system that is well suited for M2M scenarios. Our evaluation shows that d-dimensional organization model outperforms our previous solution in both speed and space requirements.**

***Keywords*—Internet of things, M2M, context information**

I. Introduction

When we think about the Internet we mostly consider servers, laptops, routers and fixed broadband that have reached almost every household. But in fact the Internet is growing as we speak. Everyday new kinds of devices (from mobile phones to environmental sensors networks) connect to the Internet, and share massive amounts of information. According to the ICT Knowledge Transfer Network (ICT KTN), the number of mobile devices is expected to increase worldwide from 4.5 billion in 2011 to 50 billions by 2020 [1].

The data generated by these devices are an untapped source of context information. This information can be used to provide added value: improve efficiency, detect abnormal conditions or advertise information. As microcosms of the Internet of Everything (IoE), cities stand to benefit the most from the untapped information shared by all these devices. Smart cities means many things to numerous people. Yet, one thing remains constant: part of being "smart" is utilizing information and communications technology and the Internet to address urban challenges. Fusing information from several sensors makes it possible to predict a driver's ideal parking spot [2], [3]. Projects such as Pothole Patrol[4] and Nericell [5] use vehicular accelerations to monitor road conditions and detect potholes. TIME (Transport Information Monitoring Environment) project [6] combines data from mobile and fixed sensors in order to evaluate road congestion in real time.

For scenarios like the previous ones to become reality, it is necessary to develop a way to manage such diverse machine made context information. One of the challenges is to store massive amounts of context information and provide a discriminative retrieval process.

Without loss of generality let us only consider pothole detection (a task of utmost importance for city officials). In an ideal context storage system, information related with road conditions should be automatically tagged with an appropriate tag. The information published by various sensors (on board of multiple vehicles from several brands) does not explicit mention road condition's, it only contains measurements related with the vehicle. It is therefore necessary to allow search with concepts instead of simple words. It is quite difficult to add these functionalities to standard full-text search engines (present in several databases).

Common definitions of context information [7], [8] does not provide any insight about its structure. In fact, each device can share context information with a different structure. Over time several representation schemes have been proposed (e.g. ContextML[9], SensorML[10] COBRA-Ont[11]). All of these representations try to standardize the process of sharing context information through several services. However, none have been widely accepted either by the academia or the industry. Usually each context-aware platform defines a context representation that suits their specific needs. This breaks compatibility between platforms and limits the quantity of context information that can be used in M2M applications. It is possible (but unlikely) that in the future a context representation standard will be widely adopted. Until then, context-aware platforms have to deal with multiple context representations.

Another important challenge is the distribution of context information. The Internet has considerably changed the scale of distributed systems. Nowadays, these systems involve thousands of entities, potentially distributed all over the world. Whose locations and behaviour may greatly vary throughout the lifetime of the system. Individual point-to-point and synchronous communications lead to rigid and static applications. As a consequence the development of dynamic large scale applications becomes cumbersome. Recently the publish/ subscribe paradigm has proven to be an adequate abstraction for large scale information dissemination. In this paradigm, subscribers specify their interest in certain event candidates. The receivers are notified every time a publisher fires and event that matches their registered interests. The strength of this event-based interaction style lies in the full decoupling

978-1-4799-6236-5/14 $31.00 © 2014 IEEE
DOI 10.1109/CyberC.2014.53

of time, space, and synchronization between publishers and subscribers.

In a previous publication [12] we proposed a context storage system. Our previous solution provides a generalized storing process and discriminative retrieval. Nonetheless, our previous solution has some drawbacks: the context organization model is not optimize for M2M scenarios, and does not support the publish/subscribe paradigm. The aim of this paper is twofold. First, we discuss the drawbacks of our previous context organization model and proposed a new one that is optimized for M2M scenarios. Second, we develop an efficient publish/subscribe mechanism based on our new organization model.

The remainder of the paper is organized as follows. In Section II we analyse how context information can be organized and define the basic requirements for context storage solutions. We discuss our previous solution's drawbacks and propose a new context organization model in Section III. Semantic-based publish/subscribe is defined and analysed in Section IV. The spatial requirements of both models is estimated in Section V. Implementation details are described in Section VI. We evaluated both approaches with a real M2M simulation. The results are discussed in Section VII. Finally, the discussion and conclusions are presented in Section VIII.

II. Context organization

Context information is an enabler for deeper and further data analysis, requiring the integration of an increasing number of information sources. As previously mentioned, nowadays no widely accepted context representation scheme exists; instead there are several approaches to deal with context information.

These approaches can be divided into three categories: adopt/create a new context representation, normalize the storing process through ontologies and accept the diversity of context representations. Previous works have adopted existing representations [9], [13], [14]. Each project defined a new optimized context representation. However, this approach imposes limits to the quantity of information that can be shared with other context-aware platforms. Later on the authors recognized that the usage of a single context representation limits the information expressiveness [14].

Another possibility would be employing ontologies to normalize the storage process. Each context representation scheme is mapped into the internal data model through an ontology [15]. This type of platform supports several context representations, yet it is necessary to define a new ontology for each representation. Defining a new ontology is a tedious task that requires human intervention. The scale of M2M scenarios make this task very difficult.

Finally, we can accept the diversity of context representation as a consequence of economic pressures, and prepare for this inevitability [12].

According to the authors [16]–[18], the best solution to characterize context information is through bottom-up characterization. Bottom-up characterization is massively dimensional, and there is no global consistency imposed by current practice. Although sensor information is not tagged by users, we can model the tagging process as keyword extraction [19]–[21]. A keyword is a sequence of one or more characteristic terms that provide a compact representation of a document's content. Ideally, keywords represent in condensed form the essential content of a document.

A context storage solution must fulfil 3 requirements: generalize storing process, discriminative retrieval and ability to scale. The first two requirements complement each other. In other words, the ideal context storage must store and accurately pinpoint any piece of context information associated with any type of sensor. The most common methods to implement discriminative retrieval are through semantic web or information retrieval system. Since semantic web methods require ontologies (which in this scenario is a disadvantage) a context database must provide a discriminative retrieval based on information retrieval systems.

The number of devices connected to the Internet is rapidly increasing, as a consequence the quantity of context information available is also increasing. A context storage solution must be robust to this increase. A database system can be distributed through several nodes in order to improve performance. Each node contains a set of the whole database (horizontal partitioning/sharding [22]).

III. Context organization model

The common definitions of context information [7], [8] are so broad that any type of information related to an entity can be considered context information. These definitions also do not provide any insight about the structure of context information. From now on we will refer to a unique piece of context information as a document.

The simplest way to model context information is through a single dimensional model (see Fig. 1). Each document is characterized with a unique key, stored in a key-value structure and indexed by an information retrieval system [12].

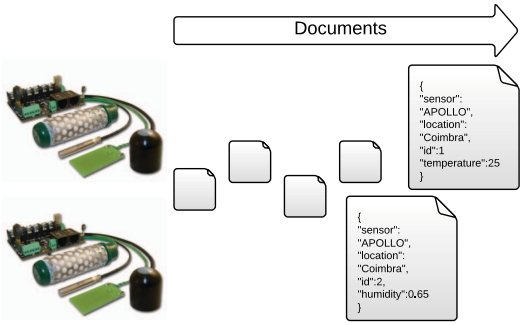


Fig. 1: Representation of a single dimensional model. The only dimension is the document identification.

This model does not take full advantage of the information retrieval system. As a consequence the single dimensional model has some disadvantages: poor scalability and semantic extraction. It is not trivial to distribute the information retrieval system through several nodes without loss of precision. As a consequence, the performance of the context storage is bounded by the information retrieval size. Consider the

following scenario where some devices published several times faster than the others. The information retrieval system is flooded with documents from the devices that publish at higher rates. As a consequence, the terms present in these messages become too common in the information retrieval system. The descriptive potential of these terms is heavily penalized.

In order to minimize these disadvantages we propose a d-dimensional model. The first dimension is always the sensor identification. Instead of storing documents independently, they are organized by sensor. The platform stores all the documents, but only needs to index the sources/sensors. The remaining $d-1$ dimensions are used to filter data from a specific source.

A d-dimensional model is as expressive as a single dimensional model within certain circumstances:

1) Each source (device/sensor) produces a continuous data stream.
2) The semantic value of the source can be extracted from a single document.

After carefully analysing several M2M scenarios, we can state that a common M2M scenario verifies these circumstances. With a single dimensional model each document is treated independently. But in reality, the majority of sensors send information periodically or when a specific event is detected. This process is better modelled as continuous data streams than a set of independent documents (circumstance 1). As such context information is better modelled with a d-dimensional model (see Fig. 2).

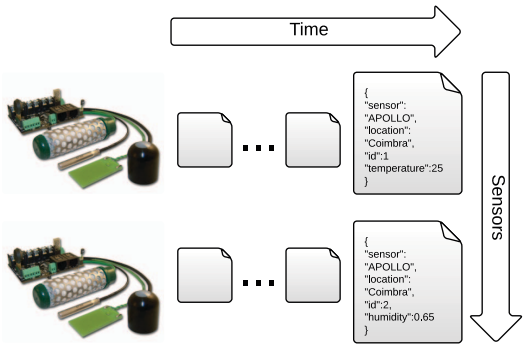


Fig. 2: Representation of a 2-dimensional model. The first and second dimensions are sensor identification and time respectively.

Data sent by the sensors is commonly represented in semi-structured format (e.g. XML, YAML, JSON, BSON). Most common semi-structured representations can be mapped into an entity-attribute-value (EAV) model [23]. The sensor is the **Entity**, what it measures are the **Attributes** and the measurements itself are the **Values**. The semantic value of a document is in the **Attributes**, the **Values** are physical measures that change over time. The **Attributes** are constant within stream, as consequence, most of the semantic value of the stream can be taken from a single document (assumption 2). Conventional information retrieval systems decomposes a document into terms without taking into account their semi-structured representation. We developed a document analyser that maps a semi-structured representation into a EAV model and only extracts the **Attributes** as characteristic terms.

In short a d-dimensional model is as expressive as a single dimensional model for M2M scenarios. At the same time a d-dimensional model improves semantic extraction and scalability. Mapping documents into a conceptual EAV model allow us to only extract terms with semantic value improving the extraction of characteristic terms. Taking into account that each source produces a continuous data stream, we only need to analyse some documents to semantically characterize the source. We have to store all the documents but only need to index information related to the sources and not all the individual documents (millions of documents but only some hundreds of sources).

Our proposed organization model analyses some documents per source and only indexes the sources, not all individual documents. There are millions of documents, but only hundreds of sources, as such the performance of the information retrieval system is only affected with the addition of new sources.

For the remaining of this paper we will only consider a 2-dimensional model (sensor identification and time). Higher dimension models only improve the selection process, do not minimize the number of sources in the information retrieval system. Without loss of generality let us consider a 3-dimensional model, composed by: sensor identification, time and location. A document is uniquely identified by a sensor id, a time stamp and a geographic location. It becomes possible to select documents based from a specific sensor, time and place. However, the sources are indexed in the information retrieval system do not change. In summary, a higher dimensional model can improve the selection in the storage component, but adds little semantic value to the information retrieval system.

IV. Semantic-based Publish/Subscribe

In this section we define semantic publish/subscribe and analyse the performance impact of a single dimension and a d-dimensional model in this component. Publish/Subscribe [24] is an important message pattern for asynchronous communications between entities. It allows for strictly decoupled communication between publishers (content producers) and subscribers (content consumers). In this paradigm, subscribers express interest in certain events. They will be notified afterwards of any events that match their registered interests. This loosely coupled approach to communication enables publish/subscribe systems to adapt to changing environments where publishers and subscribers join and leave the system without disrupting the general flow of messages.

Publish/Subscriber systems can be divided into three categories: topic-based [25], content-based [26] and type-based [27]. In **topic-based** systems the events are structured into flat or hierarchical taxonomies. Each message is characterized into a topic by the publisher. The subscriber expresses interest in one or more topics and receives all the messages published to those topics. Contrarily, in **content-based**, the messages are only delivered to a subscriber if the message content matches the constraints defined by the subscriber. Finally, **type-based** is a high level variant of the publish/subscribe paradigm which aims precisely at providing guarantees such as type safety

and encapsulation. In short, events are instances (objects) of "arbitrary" application defined types.

None of the previous publish/subscribe systems are optimized for context information. Topic-based requires predefined topics (top-down characterization/taxonomy). As discussed in Section II the best solution to characterize context information is through bottom-up characterization. At the extreme, each sensor can be a topic (only possible with a d-dimensional model). This solution implies that a user knowns the identification of each sensor he uses. In several M2M scenarios this is not feasible.

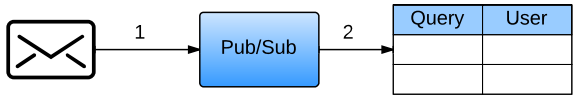
Consumers, in conventional content-based systems, subscribe events by specifying filters using a subscription language. The filters define constraints, usually in the form of name-value pairs of properties and basic comparison operators. Constraints can be logically combined to form complex subscription patterns. This strategy implies that the messages have a known representation. Context information does not have a standard representation, as such is quite difficult to evaluate content-based constraints.

Events, in type-based publish/subscribe, are objects which are instances of "arbitrary" application-defined types. This enables a closer integration of the programming language with the system. Context information is typically represented in textual format without a standard representation. As such converting context information into data types is not feasible in many M2M scenarios.

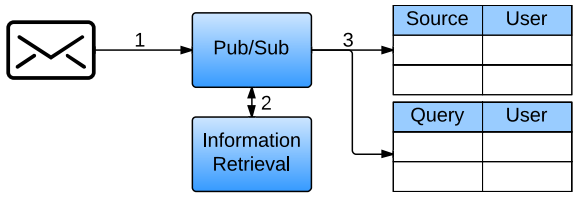
As a counterpart to these approaches, we propose a semantic-based publish/subscribe system. It is a specialization of a content-based solution, that allows users to subscribe semantic queries. Similar to a content-based solution, a user subscribes messages based on their content. However, neither the user nor the system have to known the message's structure, fields or values to write a semantic query. Alternatively, our solution can be understood as a topic-based system where the topics are dynamical created based on the user queries and the existing sources. In short, a user expresses interest in concepts (characteristic terms) instead of filters (name-value pairs of properties).

The architecture of the semantic-based solution depends greatly on the context organization model. Let us consider the models discussed in Section III: single dimensional model and d-dimensional model. Both solutions require a table to hold the subscriptions, in the form of query-user pairs. A d-dimensional model organizes documents into streams based on the source/sensor. We take advantage of this property and transform each semantic query in a set of sources/sensors. This information is stored in another table, in the form of source-user pairs. Considering that each source is a topic, a semantic-based solution can be reduced to a topic-based one.

Fig. 3 shows the steps necessaries to complete a subscription in a solution based on a single-dimensional model and a d-dimensional model respectively. The first solution receives a subscription (1) and stores it in the table (2). On the other hand, a solution based on a d-dimensional model, resolves the semantic query into a set of sources (Fig. 3b). The system receives a subscription (1), queries the information retrieval system to resolve the semantic query into a set of sources (2). Finally, it stores the subscription and the sources into the respectively tables.



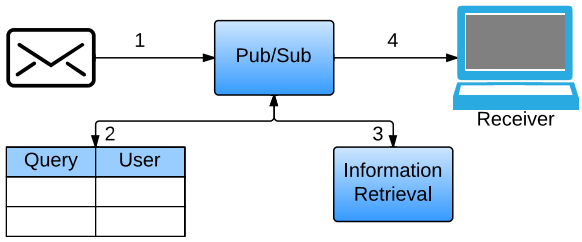
(a) Single-dimensional model



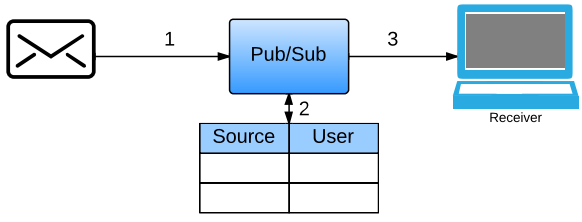
(b) N-dimensional model

Fig. 3: Steps necessary to complete a subscription in a semantic-based solution.

Fig. 4 shows the necessary steps to complete a publish operation in a solution based on a single-dimensional model and a d-dimensional model respectively. The first solution can not resolve semantic queries during subscriptions. As a consequence, it has to communicate with the information retrieval system during each publish (see Fig. 4a). The publish/subscribe system receives a publish message (1) and retrieves all the semantic queries from the table (2). After this it communicates with the information retrieval system to match the queries against the message (3), and finally forwards the message to the respective receivers. However, the second solution (see Fig. 4b) only requires a table lookup (2) in order to forward the message to the receivers (3).



(a) Single-dimensional model



(b) D-dimensional model

Fig. 4: Steps necessary to complete a publish in a semantic-based solution.

Both approaches rely on the information retrieval system

to resolve semantic queries. The performance of these approaches depends greatly on the information retrieval system. As previously discussed a d-dimensional model improves the scalability and the semantic extraction of the system. In short, the publish/subscribe solution based on a d-dimensional model has all the advantages associated with this model. Furthermore, during a subscription the semantic query is transformed into a set of relevant sources. As such, a publish operation only requires the source table. Commonly, in a M2M scenario there is a greater number of publish operations than subscribe operations. In other words, the pre-processing step improves the performance for real world applications.

V. Context storage space requirements

Within this section we compared the space requirements of a context storage solution based on a single-dimensional and a d-dimensional model. The size of the storage solution can be roughly estimated as the sum of the sizes of the information retrieval (IR), the storage (S) and the publish/subscribe system (PS):

$$Size(CS) = Size(S) + Size(IR) + Size(PS) \quad (1)$$

We do not take into account the publish/subscribe' size, since it only makes for a small fraction of the context storage space requirements. As such, equation 1 can be reduced to:

$$Size(CS) = Size(S) + Size(IR) \quad (2)$$

The information retrieval system, in a single-dimensional model, contains all the documents. But the storage system, in a d-dimensional model, contains additional fields (one field for each dimension). We want to identify when a d-dimensional model is more spatially efficient than a single-dimensional one. This can be achieve by solving the following relation:

$$Size(CS_{Single}) \geq Size(CS_D) \quad (3)$$

Based on equation 2 we can derive the size of a solution based on a single and a d-dimensional model:

$$Size(CS_{Single}) = \sum_{i=1}^{N}(size(uid_i) + size(doc_i)) + \sum_{i=1}^{N}\sum_{j=1}^{T} size(term_{ij}) \quad (4)$$

Where N and T represent the number of documents and characteristic terms respectively. The storage system uses is a tabular database that uses a table to store context information. The table holds pairs of unique identifier and documents. As such the size of this component is the area of the table. The majority of information retrieval use a term-document matrix to compute the similarity between documents and queries. In a term-document matrix specific type of co-occurrence matrix, each row represents a unique term and each column represents a document. The size of the co-occurrence matrix can be used as a rough estimate for the size of the index.

The size of a d-dimensional based solution is estimated with the following equation:

$$Size(CS_D) = \sum_{i=1}^{N}\left(size(uid_i) + size(doc_i) + \sum_{j=1}^{D-1} size(field_j)\right) + \sum_{i=1}^{K}\sum_{j=1}^{T} size(term_{ij}) \quad (5)$$

Where N, T, K, D represent the number of documents, characteristic terms, sources/sensors and dimensions respectively. This equation differs from the previous only in two aspects. It accounts for additional dimensions in the storage system, and the information retrieval system only index the sensors/ sources.

After applying equations 4 and 5 into inequality 3 and reducing, we obtain the following result:

$$\sum_{i=1}^{N-K}\sum_{j=1}^{T} size(term_{ij}) \geq \sum_{i=1}^{N}\sum_{j=1}^{D-1} size(field_j) \quad (6)$$

In short, a d-dimensional model is more spatially efficient when the size of the additional fields is smaller than the size of all the indexed documents. Commonly the additional are timestamps, geohash, latitudes, longitudes, etc. In other words, these fields can be store in a integer/float. The $termij$ represents the weight of the term j in the document i. This element can also be stored as float. We can continue to solve the inequality replacing $size(term_{ij})$ and $size(field_j)$ with a constant C:

$$\sum_{i=1}^{N-K}\sum_{j=1}^{T} C \geq \sum_{i=1}^{N}\sum_{j=1}^{D-1} C \quad (7)$$

$$(N-K) \times T \times C \geq N \times (D-1) \times C \quad (8)$$

$$NC + NTC - NDC \geq KTC \quad (9)$$

$$N \geq K \times \frac{T}{1+T-D} \quad (10)$$

A d-dimensional model has a finite number of dimensions, usually 2 or 3. The number of characteristic terms is potentially unbounded. As the number of characteristic terms grows, the term $\frac{T}{1+T-D}$ tends to 1: $\lim_{T\to\infty} \frac{T}{1+T-D} = 1$ Thus, a d-dimensional model is more spatially efficient when the documents' number is greater than the sources/sensors. In a common M2M scenario the documents' number far exceeds the sources. Hence, a d-dimensional model is commonly more spatially efficient than a single-dimension one for M2M scenarios. It's worth pointing out that we did not take into account compression mechanisms. However, these mechanisms should have an equal effect in both models, thus they have little impact in our comparison.

VI. Implementation

In this section we discuss important details about our solution. The software architecture of the solution is almost

the same for the single-dimensional and 2-dimensional model. Our context storage solution can be divided into four different components as depict in Fig. 5. The storage and index components store and index context information respectively. Furthermore, the pub/sub component implements the semantic-based publish/subscribe system described in this paper. Finally, the router communicates with the remaining components in order to fulfil each operation. All of them communicate with each other through message passing.

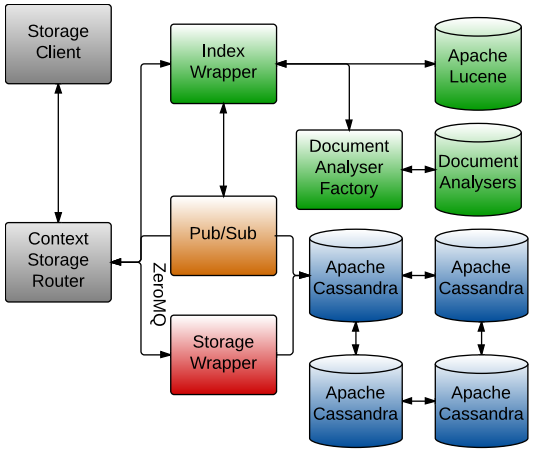


Fig. 5: Propose context storage architecture.

The components communicate with each other using the ZeroMQ[1] socket library. This library supports several transportation methods: TCP sockets, inter-process and inter-thread communication. Messaging passing allows the application to be distributed through several machines and each component can be written in any programming language, without being restricted by the router component. This strategy is then specially suitable for the diversity of environments in M2M applications.

The index component is mainly an information retrieval system. It was prototyped in Java, using Apache Lucene[2] at its core. This component was developed with special attention to parallelism. The IndexWriter class was expanded to support periodical commits (safe store in the disk) with a background thread. The component also uses near-real-time search[3]. For the 2-dimensional model we also developed a custom document analyser that maps JSON documents into a EAV model and extract the semantic value of the attributes.

The storage component is mainly a tabular database responsible for storing all the documents. It was prototyped in Java, using Apache Cassandra[4] as its core. Apache Cassandra is one of the fastest tabular databases currently available, initially developed by Facebook and inspired by Amazon's Dynamo [28]. Cassandra is designed to handle big data workloads across multiple nodes with no single point of failure. The context information is stored in a single table. The single-dimensional model uses a table with two columns: the first column holds a unique identifier, and the last column holds the document. The 2-dimensional model uses the same table with an additional column to hold the timestamp.

The pub/sub component was prototyped in Java, using Apache Cassandra to store the subscriptions and ZeroMQ library for communication. Both context organization models require a table with two columns to hold the semantic subscription: the first column holds the query, and the last column holds the user unique identification. The 2-dimensional model requires an additional table to hold the resolution of the semantic queries into a set of sources. This table has two columns: the first and the second column holds the source and user unique identification.

VII. Performance Evaluation

In this section we evaluate the performance of two context storage solutions. These solutions are based on a single-dimensional and a 2-dimensional models respectively.

We developed a simulation based on the APOLLO project[5]. The APOLLO project's main objective is the development of a platform that supports new scenarios in the area of M2M communications. This platform uses the storage solution described in this paper and was instantiated in two different scenarios: greenhouse monitoring and pothole detection. The first scenario was composed of 7 sensor nodes that collected environmental data regarding a green house. The pothole detection scenario consists of identifying potholes on the road based on the vibrations of the vehicle (use-case similar to the Pothole Patrol project [4]). In this scenario motorized vehicles have a sensor node that measures the acceleration, geographic location and the speed. Reports containing detected potholes are generated periodically. The sensor sends information only when the vehicle has a speed greater than 2.5 m/s.

Our simulation consists of a pothole detection service, and two classes of sensor node: vehicular and greenhouse sensor nodes. The simulation lasts for 3 months (90 days). During this period the sensors nodes generate almost 2 million documents, which corresponds to 3 gigabytes of information. The pothole detection service communicates with the storage solution through two distinct interaction paradigms: publish/subscribe and request/reply. Every 3 days a new report is generated.

TABLE I summarizes the parameters that where used in our simulation. The publish rates were computed based on the APOLLO project information. For efficiency reasons we accelerated the simulation 720 times (it took 3 hours instead of 90 days). This increase in speed can be interpreted as a load effect of 720 times more sensor nodes (it has a similar effect) and thus also illustrates the system's scalability.

The simulation ran in a desktop computer with the following specifications: 8 GB of memory RAM and 8 CPUs with a 1.6 GHz clock speed. The machine had a Linux operating system (kernel version 3.14.3), Apacha Cassandra (2.0.7), ZeroMQ socket library (4.0.4) and Apache Lucene (4.8).

At turns we evaluate both storage solutions and both interaction paradigms. We measure the duration of two distinct

[1] http://www.zeromq.org/

[2] lucene.apache.org/core/

[3] blog.mikemccandless.com/2011/06/lucenes-near-real-time-search-is-fast.html

[4] cassandra.apache.org

[5] http://atnog.av.it.pt/projects/apollo

TABLE I: Simulation's parameters

Simulation parameters	
Vehicles publish rate	137 seconds
Greenhouse publish rate	120 seconds
Pothole reports	every 3 days
Duration	3 month (90 days)
Warm up	1 day

operations: publish context information (write operation) and gather a dataset (read operation) from the storage solution. The duration of a read operation through the publish/subscribe paradigm consists only in the time a messages takes to reach the receiver.

On the other hand, a read operation through the request/ reply is decomposed in two different operations: a search and a select operation. A read operation consists in finding the correct sensor nodes (search) and retrieving the corresponding documents from a specific time period (select). As such the duration of a read operation is the sum of the search and select operations.

The average operation's time for request/reply and publish/ subscribe interactions are summarized in TABLE II and III respectively. Our solution based on a 2-dimensional model outperforms a storage system based on a single-dimensional model. A 2-dimensional model yields a slight performance improvement for write operations (a speedup of 1.26 and 59.88 for request/reply and publish/subscribe respectively). Yet the greatest performance increase is seen on the read operations (a speedup of 534.47 and 1015550.20 request/reply and publish/ subscribe respectively).

TABLE II: Request/Reply performance evaluation

Organization Model	Write Operation	Read Operation
Single-Dimension	2.06 $\pm$ 15.00	1519043.57 $\pm$ 1021511.72
Two-Dimension	1.64 $\pm$ 1.09	2842.13 $\pm$ 439.96
SpeedUp	1.26	534.47

TABLE III: Publish/Subscribe performance evaluation

Organization Model	Write Operation	Read Operation
Single-Dimension	70.06 $\pm$ 48.21	2528719.99 $\pm$ 1358701.8
Two-Dimension	1.17 $\pm$ 0.96	2.49 $\pm$ 1.21
SpeedUp	59.88	1015550.20

We computed partials read and write averages 10 times during the simulation Fig. 6 shows the partial read averages for both models and interaction paradigms. The performance of the 2-dimensional model is stable during all the simulation. However, the performance of the single-dimensional model degrades as the simulation progresses. This model does not provide any mechanism to group or filter documents. Thus, an entity has to iterate though all the matches of a semantic search in order to find the relevant documents. Since the information retrieval system contains all the documents a read operation becomes quite inefficient. It's worth pointing out that the solution based on a single-dimensional took more than 90 days to generate the reports. Therefore, Fig. 6 only contains the partial reads that took place inside the 90 days window.

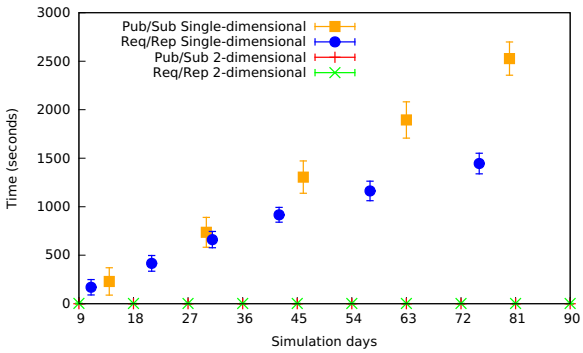


Fig. 6: Partial read averages

Fig. 6 shows the partial write averages for both models and interaction paradigms. There is no clear performance degradation in both models, the partial write averages are stable during the simulation. Yet, the variance of the single-dimension model increases as the simulation advances. This indicates that the write performance becomes unstable over time. On the other hand, the variance of the 2-dimensional model is stable during the simulation. In fact the variation is so small that is almost imperceptible in the graph.

Let us consider a single dimensional model. Read operations depend greatly in the information retrieval system. Therefore, each read operation stresses this component. As a consequence the information retrieval systems takes longer to reply which in turn affects the performance of the write operations.

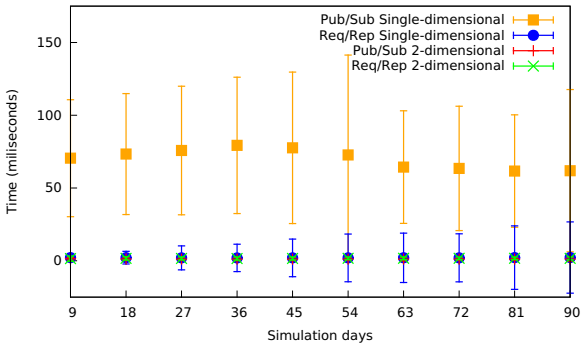


Fig. 7: Partial write averages

VIII. Discussion and Conclusions

Within this paper, we discuss the drawbacks of our previous context organization model and proposed a new one that is optimized for M2M scenarios. We also develop an efficient publish/subscribe mechanism based on our new organization model.

We analytically evaluated the spatial efficient of both approaches. Based on our analysis, a d-dimensional model is more spatially efficient when the documents' number is greater than the sources/sensors. In short, a d-dimensional model is commonly more spatially efficient than a single-dimension one for M2M scenarios. We also used a simulation of a real M2M scenario to evaluate the performance of both solutions. The d-dimensional model outperforms a solution based on a single-dimensional model. In short, our d-dimensional model performs better overall for M2M scenarios.

The d-dimensional model takes full advantage of the information retrieval system. It improves the solution's scalability and semantic extraction, by carefully grouping documents into sources. However, it is not trivial to distribute this system through several nodes without loss of precision. As a consequence, the performance of the context storage is bounded by the information retrieval system. In future publications we will address this issue.

Acknowledgement

This work has been partially funded by the Portuguese Innovation Agency/National Strategic Reference Framework (AdI/QREN) under grant agreement No. 2011/021580 (APOLLO project) and by project Cloud Thinking (CENTRO-07-ST24-FEDER-002031), co-funded by QREN, "Mais Centro" program.

References

[1] U. F. I. S. Group, "Future internet report," ICT Knowledge Transfer Network, Tech. Rep., May 2011.

[2] T. Rajabioun, B. Foster, and P. Ioannou, "Intelligent parking assist," in *Control Automation (MED), 2013 21st Mediterranean Conference on*, June 2013, pp. 1156–1161.

[3] J. K. Suhr and H. G. Jung, "Sensor fusion-based vacant parking slot detection and tracking," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 15, no. 1, pp. 21–36, February 2014.

[4] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan, "The pothole patrol: Using a mobile sensor network for road surface monitoring," in *Proceedings of the 6th International Conference on Mobile Systems,Applications, and Services*, 2008, pp. 29–39.

[5] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: Rich monitoring of road and traffic conditions using mobile smartphones," in *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, ser. SenSys '08. New York, NY, USA: ACM, 2008, pp. 323–336.

[6] J. Bacon, A. Bejan, A. Beresford, D. Evans, R. Gibbens, and K. Moody, "Using real-time road traffic data to evaluate congestion," in *Dependable and Historic Computing*, ser. Lecture Notes in Computer Science, C. Jones and J. Lloyd, Eds. Springer Berlin Heidelberg, 2011, vol. 6875, pp. 93–117.

[7] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in *Proc. of the 1st international symposium on Handheld and Ubiquitous Computing*, 1999, pp. 304–307.

[8] T. Winograd, "Architectures for context," *Hum.-Comput. Interact.*, vol. 16, no. 2, pp. 401–419, December 2001.

[9] M. Knappmeyer, S. L. Kiani, C. Frà, B. Moltchanov, and N. Baker, "Contextml: a light-weight context representation and context management schema," in *Proc. of the 5th IEEE international conference on Wireless pervasive computing*, 2010, pp. 367–372.

[10] M. Botts and A. Robin, "Opengis sensor model language (sensorml) implementation specification," OGC, Tech. Rep., July 2007.

[11] H. Chen, T. Finin, and A. Joshi, "An ontology for context-aware pervasive computing environments," *The Knowledge Engineering Review*, vol. 18, pp. 197–207, Setember 2003.

[12] M. Antunes, D. Gomes, and R. Aguiar, "Context storage for m2m scenarios," in *Proc. ICC 2014*, 2014.

[13] B. Moltchanov, M. Knappmeyer, C. A. Licciardi, and N. Baker, "Context-aware content sharing and casting," in *Proceedings of ICIN*, 2008.

[14] T. Mota, N. Baker, B. Moltchanov, R. Ioanna, and K. Frank, "Towards pervasive smart spaces: A tale of two projects," in *Future Network & Mobile Summit 2010. The Second International Workshop on Information Quality and Quality of Service for Pervasive Computing in Conjunction with IEEE PERCOM 2010*, 2010.

[15] P. Lopes and J. L. Oliveira, "Coeus: Semantic web in a box for biomedical applications," *Journal of Biomedical Semantics*, vol. 3, no. 1, p. 11, 2012.

[16] C. Shirky, "Ontology is overrated: Categories, links, and tags," http://shirky.com/writings/ontology_overrated.html, May 2005, accessed: 22-07-2013.

[17] G. Avram, "At the crossroads of knowledge management and social software," *Electronic Journal of Knowledge Management*, vol. 4, no. 1, pp. 1–10, January 2006.

[18] T. Gruber, "Ontology of folksonomy: A mash-up of apples and oranges," *International Journal on Semantic Web and Information Systems*, vol. 3, no. 2, pp. 1–11, 2007.

[19] A. Hulth, "Improved automatic keyword extraction given more linguistic knowledge," in *Proc. of the 2003 conference on Empirical methods in natural language processing*, 2003, pp. 216–223.

[20] R. Mihalcea and P. Tarau, "Textrank: Bringing order into texts," in *Conference on Empirical Methods in Natural Language Processing*, 2004.

[21] S. Rose, D. Engel, N. Cramer, and W. Cowley, *Automatic Keyword Extraction from Individual Documents*. John Wiley and Sons, Ltd, March 2010, pp. 1–20.

[22] S. Ceri, M. Negri, and G. Pelagatti, "Horizontal data partitioning in database design," in *Proc. of the 1982 ACM SIGMOD international conference on Management of data*, 1982, pp. 128–136.

[23] P. M. Nadkarni, L. Marenco, R. Chen, E. Skoufos, G. Shepherd, and P. Miller, "Organization of heterogeneous scientific data using the eav/cr representation," *Journal of the American Medical Informatics Association*, vol. 6, no. 6, pp. 478–493, 1999.

[24] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Comput. Surv.*, vol. 35, no. 2, pp. 114–131, June 2003.

[25] B. Oki, M. Pfluegl, A. Siegel, and D. Skeen, "The information bus: An architecture for extensible distributed systems," *SIGOPS Oper. Syst. Rev.*, vol. 27, no. 5, pp. 58–68, December 1993.

[26] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Design and evaluation of a wide-area event notification service," *ACM Trans. Comput. Syst.*, vol. 19, no. 3, pp. 332–383, Aug. 2001.

[27] P. Eugster, "Type-based publish/subscribe: Concepts and experiences," *ACM Trans. Program. Lang. Syst.*, vol. 29, no. 1, pp. 1–50, January 2007.

[28] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: Amazon's highly available key-value store," *SIGOPS Oper. Syst. Rev.*, vol. 41, no. 6, pp. 205–220, October 2007.